Building and Testing Entropy Sources for Cryptography

Think openly, build securely

WPI ECE

October 6, 2021

Dr. Markku-Juhani O. Saarinen

Senior Cryptography Engineer, PQShield Ltd.

Outline

Entropy Sources for Cryptography

Touching on three related topics, time permitting:

1. Introduction to Entropy Sources. Current standards and terminology, physical noise sources, other necessary components.

2. RISC-V Entropy Source Interface (ISA) currently being ratified.

3. How to get an Entropy Source Module into your RISC-V SoC.

Four Statistical Tests

Computation was manual (1938), punch card (1948)

Four Tests for Local Randomness.

24. For practical purposes in deciding whether a given set is locally random, we have found that the following four tests are useful and searching. They are, however, not sufficient to establish the existence of local randomness, although they are necessary.

Procedure. Four methods of examining the million digits for randomness have been utilized. All tables and computations were accomplished by use of I.B.M. equipment.

Not Generating Secrets?

You can use such “Statistical Tests of Randomness”

- A really traditional (M. G. Kendall and B. Babington Smith, 1938) approach consists of four tests: *Frequency test, Serial test, Poker test, Gap test.*

- A version was adopted by RAND Corp. (B. Brown, 1948) for the production of the “million digits” book: *Frequency Test, Poker Test, Serial Test, Run Test.*

- FIPS 140-1 (1994) and FIPS 140-2 (2001) until December 2002: *Monobit test, Poker Test, Runs Test, Long Runs test.* .. Still survives as a part of CC / AIS-31.

- Suites such as NIST SP 800-22, Diehard are no longer used in cryptographic evaluation. Fine for testing “fast PRNGs” for Monte Carlo simulations etc.

Four Statistical Tests

In our memories, 1938-2002

Change Notice 2

FIPS PUB 140-2, SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Gaithersburg, MD 20899

DATE OF CHANGE: 2002 December 03

TITLE: Random Number Generator Requirements

Federal Information Processing Standard
specifies the security requirements that w
security system protecting sensitive but u
information).

~~*Statistical random number generator tests.* If statistical random number generator tests are required (i.e., depending on the security level), a cryptographic module employing RNGs shall perform the following statistical tests for randomness. A single bit stream of 20,000 consecutive bits of output from each RNG shall be subjected to the following four tests: monobit test, poker test, runs test, and long runs test.~~

This change notice provides corrections to the requirements for random number generator used by cryptographic modules. These corrections involve paragraphs 4.7.1 and 4.9.1 of FIPS 140-2. Table 1 – *Summary of security requirements* has also been corrected and involves the random number generator requirements.

Generating Secrets?

Things have changed

- We're not in the 1930s. Information theory has been invented. There's good cryptography and computers have AES instructions.

- FIPS 140-2 testing of pseudorandom output was obviously pretty pointless. Think of AES256-CTR keyed with the **time of day** -- `time(NULL)` -- will obviously have perfect output statistics, but no security due to the lack of *secret entropy* in the time() input. 🤦

- Complete overhaul of FIPS 140 (and NIAP) random since 2010s->

DRBG = Cryptographic PRNG

With reseeding mechanisms, security features

- DRBGs (Deterministic Random Bit Generators) based on ciphers and hashes do not allow their ***secret state*** to be determined from output.

  This can be shown to be equivalent to breaking AES or SHA-2/3.

- Features: Backtracking resistance against seed compromise (you can't determine previous DRBG outputs even if you know current state).

- *DRBGs are not the topic of this talk.* NIST SP 800-90A defines the standard ones: CTR_DRBG, Hash_DRBG, HMAC_DRBG. Other de facto DRBGs in Linux Kernel /dev/random, Apple's Fortuna, in OpenSSL..

Standard Terminology

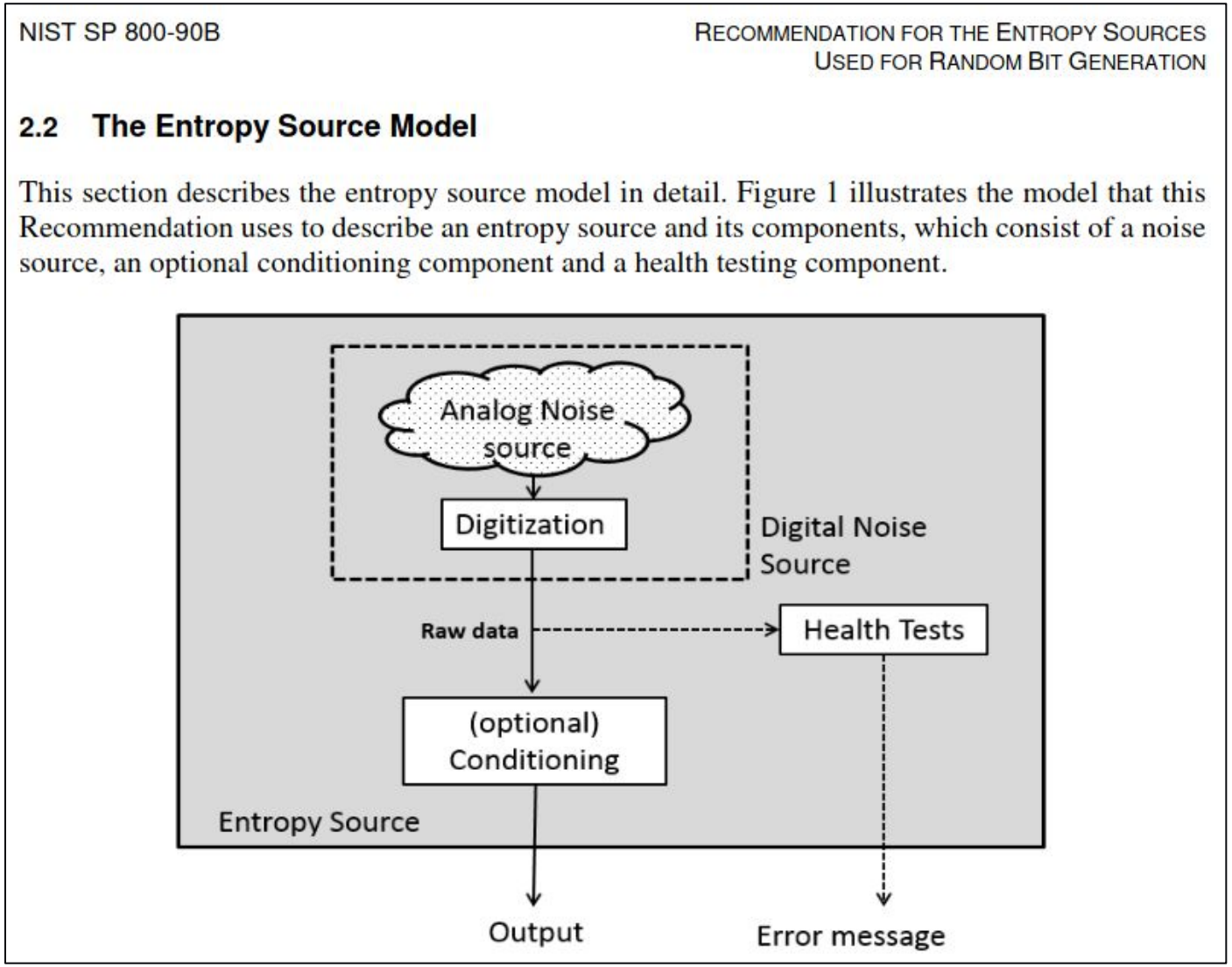
Mandatory now: Use a RBG with an Entropy Source

- **RBGs** (not “RNGs”) are used to generate **SSPs** (Sensitive Security Parameters) such as secret cryptographic keys.

- A **DRBG** is a good **RBG** if seeded appropriately with entropy; this is defined in NIST/FIPS. (The term “PRNG” is more fuzzy.)

- An **Entropy Source** contains a well-understood random **Noise Source**, and must also have **health tests** that detect failures. *Not necessarily uniform, but has a guaranteed output entropy rate.*

What is an Entropy Source? (USA)



NIST SP 800-90B

RECOMMENDATION FOR THE ENTROPY SOURCES USED FOR RANDOM BIT GENERATION

2.2 The Entropy Source Model

This section describes the entropy source model in detail. Figure 1 illustrates the model that this Recommendation uses to describe an entropy source and its components, which consist of a noise source, an optional conditioning component and a health testing component.

Entropy Source contains a well-understood **Noise Source**.

Health tests are a **mandatory** part of an entropy source.

Conditioning is optional but can be inside the box.

Output distribution has ***known lower bound on entropy rate.***

AIS 20 / AIS 31 Terms (BSI, CC)

13 **Cryptographic post-processing**

A post-processing algorithm that generates the internal numbers of a TRNG by means of a cryptographic mechanism

14 **das-random number**

Bit string that results directly from the digitization of analogue noise signals (das) in a physical RNG. Das-random numbers constitute a special case of raw random numbers.

NOTE: Assume, for instance, that a PTRNG uses a Zener diode. Regular comparisons of the (amplified) voltage (analogue signal) with a threshold value provide values 0 and 1, which may be interpreted as das-random numbers. In contrast, for ring oscillators on FPGAs it is not obvious how to define the analogue signal. At least in the true sense of the word it may be problematic to speak of 'das random number' in this context.

NOTE: In [AIS31An] for physical RNGs the term 'das-random number' was consistently used. Apart from concrete examples in this document we use the more general term 'raw random number' for both physical and non-physical true RNGs.

15 **Deterministic RNG**

AIS 31 is used in intl. Common Criteria certifications; was clearly better than FIPS evaluation until FIPS 140-3 & SP 800-90B (2018).

AIS 31 says **Post-processing** when FIPS/90B says **Conditioning.**

Das-random or **raw random** are used for noise source output.

Uses TRNG / PTRNG acronyms.
(Harmonization expected.)

Physical Noise Source

What is Required?

It doesn't need to be statistically "perfect" but it must be very well understood.

- ✅ A stochastic model (or a heuristic argument) that explains why the noise source output is from a random, rather than pseudorandom (deterministic) process.
- ✅ A complete physical model is not necessary, but entropy needs to be justified on the technology node (Semiconductor process? Temperature? Freq? etc.)
- ✅ The noise source is not externally observable (e.g., strongly correlated with the time of day, external power or emissions, or otherwise public or predictable).

Example: For a *ring oscillator* one could show that the source derives an amount of physical entropy from local, spontaneously occurring Johnson-Nyquist thermal noise.

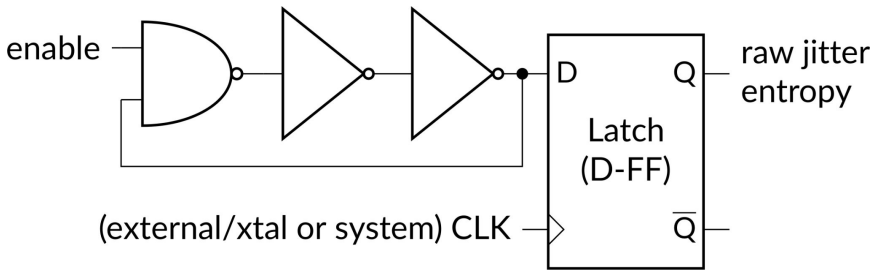
Physical Noise Sources

Need well-understood Entropy: Stochastic Models etc.

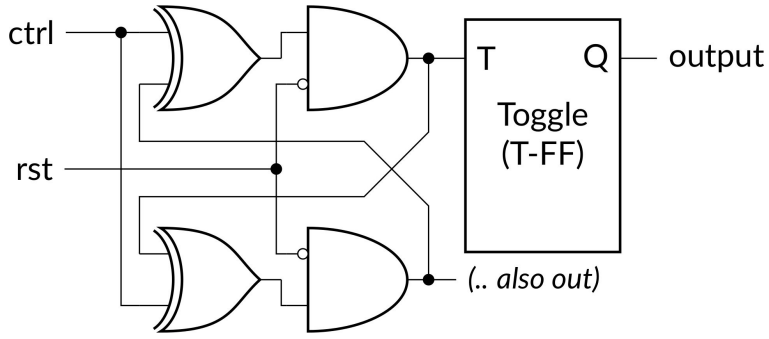
- **Ring Oscillators** are common choices. Can be built with standard cell libraries: Cheap and small area (as little as few thousand NAND2 gates).

- RISC-V ES spec allows any physical noise source suitable for 90B or AIS-31.

- Note: Post-Quantum Crypto (PQC) doesn't need "quantum" noise sources.



A ring oscillator with 3 inverters in a free running loop.



A temporarily oscillating "metastable" TERO configuration.

Low Bit Rates are Usually Okay

Remember that the Entropy Source just seeds the DRBG

Features

The TRNG generates a random bit stream.

The TRNG core has the following key features:

- Produces 10K bits/second of entropy when core is running at 200MHz.
- Includes an internal entropy source that is based on a chain of digital inverters. The inverter cells are taken from a standard cell library. No special cells are required.
  - Odd number of inverters, leading to continuous oscillation (while active).
- Built-in hardware tests for auto correlation and *Continuous Random Number Generation Testing* (CRNGT) as required by the following standards:
  - FIPS 140-2, *Security Requirements for Cryptographic Modules*.
  - AIS-31, *Functionality Classes and Evaluation Methodology for True Random Number Generators*.
- AMBA APB2 slave interface.

ARM's "TrustZone TRNG" is likely to be the most popular Random Number Silicon IP.
*In current (2020-) FIPS 140-3 terminology it is probably an **Entropy Source**.*

Stochastic Models

Justification: Gaussian Jitter from Physical Sources

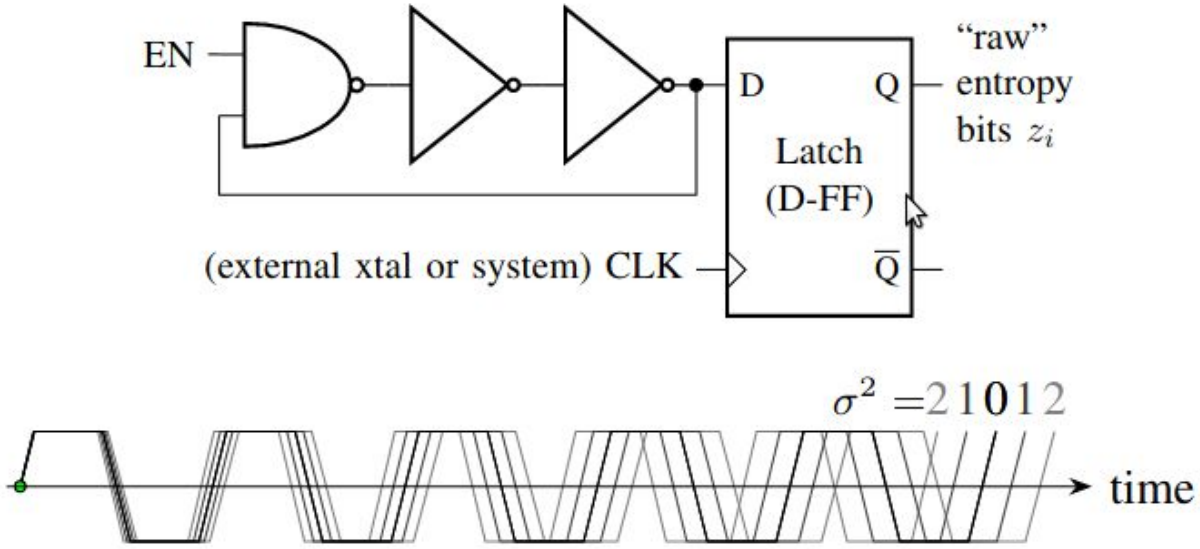


Fig. 1. A ring oscillator consists of an odd (here $N = 3$) number of inverters connected into a free-running loop. The output is sampled using an independent reference clock, such as a crystal oscillator. Transition times are affected by jitter (largely from Johnson-Nyquist thermal noise), whose accumulation causes samples to become increasingly unpredictable.

An example of a detailed physical model for ring oscillator phase noise and jitter is provided by Hajimiri et al [13]–[15], which we recap here. The randomness of timing jitter has a strongly Gaussian character. The jitter accumulates in the phase difference against the reference clock, with variance σ_t^2 growing almost linearly from one cycle to the next.

Under common conditions, the transition length standard deviation (uncertainty) σ_t after time t can be estimated for CMOS ring oscillators as (after [14, Eqns. 2.6,5.18]):

$$\sigma_t^2 = \kappa^2 t \approx \frac{8}{3\eta} \cdot \frac{kT}{P} \cdot \frac{V_{DD}}{V_{\text{char}}} \cdot t \qquad (1)$$

In this derivation of physical jitter κ^2 we note especially the Boltzmann constant k and absolute temperature T; other variables include power dissipation P, supply voltage V_{DD}, device characteristic voltage V_{char}, and a proportionality constant $\eta \approx 1$. The number of stages (N) and frequency f affect power P via common dynamic (switching) power equations.

M.-J. Saarinen, *"On Entropy and Bit Patterns of Ring Oscillator Jitter" (2021)* https://arxiv.org/abs/2102.02196

Stochastic Models

Justification: Gaussian Jitter to Entropy

- With a model like “Gaussian Thermal Jitter” σ^2 you can derive a distribution for output bits. This is the stochastic model.

- Stochastic model is required for AIS-31, preferred for SP 800-90B (heuristic arguments may still be enough for a FIPS 140-3 ENT).

- The stochastic model, together with measurements is used to justify entropy content of noise source output. This is a key part of the Test Lab “entropy source report” in security certification.

Entropy Metrics

Shannon Entropy > Min-Entropy (Rényi Entropies)

Shannon Entropy in AIS-31: Traditional "Compression Entropy"

$$H_1(Z_n) = -\sum_z p_z \log_2 p_z$$

Min-Entropy of SP 800-90B: "Guessing Entropy"

$$H_\infty(Z_n) = \min_z(-\log_2 p_z) = -\log_2(\max_z p_z)$$

Min-entropy is always smaller (or equivalent) to Shannon entropy. Can be a lot smaller! *Has different combining and "algebra" rules.*

Entropy From Model

Reports can borrow formulas from Academic Literature, but..

Are the formulas really sane and applicable?

This is complex-looking thing is sometimes seen in submissions:

$$H \geq H\left(s(\Delta t) \mid \varphi(0)\right) = 1 - \frac{4}{\pi^2 \ln(2)} e^{-4\pi^2 Q} + O\left(e^{-6\pi^2 Q}\right). \qquad (14)$$

(From: M. Baudet et al. *"On the Security of Oscillator-Based Random Number Generators"* J. Cryptol. (2011) 24: 398-425.)

The formula implies Entropy $H_1 > 0.4$ with **no Jitter** (Q = 0)..

SP 800-90B Entropy Estimators

A Big Set of Estimators, take the lowest value

- There is also a standard set of **entropy estimators** for the noise source and entropy output (both IID and non-IID versions).

- Lowest min-entropy estimate is the one overall result.

https://github.com/usnistgov/SP800-90B_EntropyAssessment

1. *The Most Common Value Estimate*
2. *The Collision Estimate*
3. *The Markov Estimate*
4. *The Compression Estimate*
5. *t-Tuple Estimate*
6. *Longest Repeated Substring (LRS)*
7. *The MultiMCW Prediction Estimate*
8. *The Lag Prediction Estimate*
9. *The MultiMMC Prediction Estimate*
10. *The LZ78Y Prediction Estimate ..*

Health Testing Circuitry

Integral part of any Entropy Source

- SP 800-90B mandates circuitry in that monitors output bits and performs **Startup**, **Continuous**, and **On-Demand** Health Tests.

- Explicitly approved continuous health tests: Repetition Count Test (**RCT**), Adaptive Proportion Test (**APT**). Fairly simple fixed circuitry. Failure bounds are be derived from entropy estimates/claims.

- Vendor-defined tests. These should be derived from expected failure modes of the specific physical noise source design.

- **Example**: Ring Oscillator jitter is highly dependent on temperature; health tests must catch temperature anomalies, prevent output.

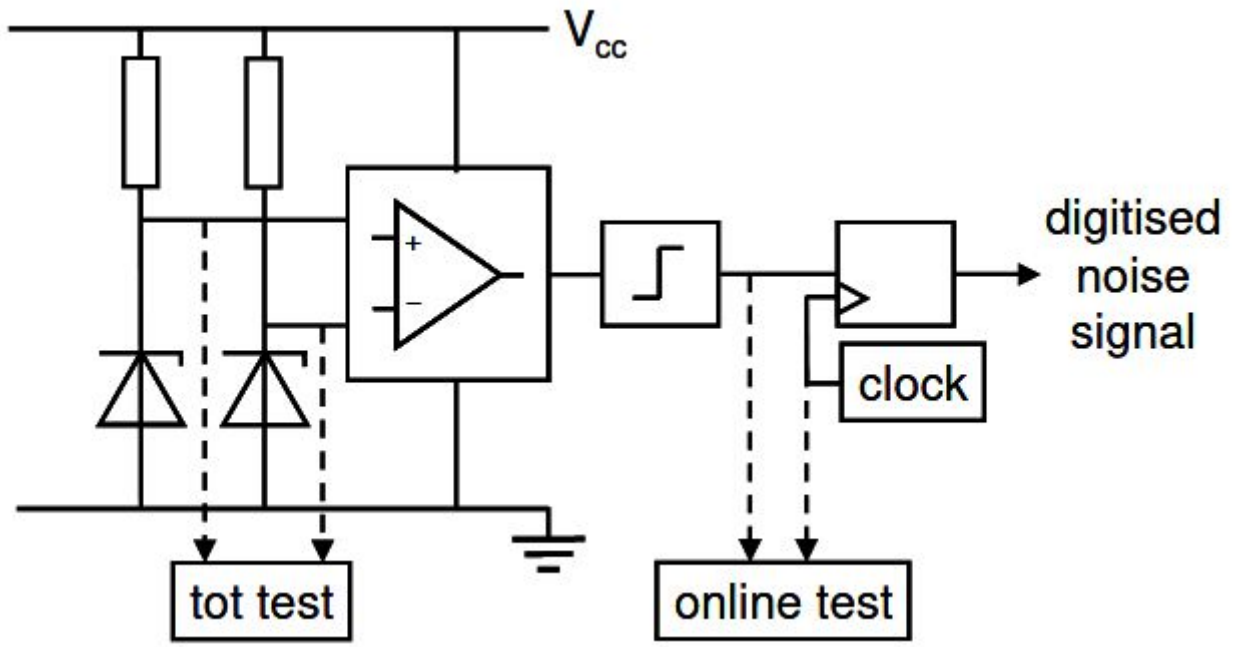
Conditioners

Integral part of any Entropy Source

- **Conditioners** condense raw noise source output into shorter bit strings with higher and/or more consistent entropy rate.
- Optional. Can be a part of the entropy source or external (90C).
- **Vetted conditioners:** Things like NIST Hash functions (SHA-2/3), HMACs and some AES-based constructions are *pre-approved*.
- **Non-Vetted conditioners:** Mathematical entropy extractors; von Neumann debiaser and other lightweight conditioners need testing but are fine as an intermediate post-processing step.

Other Electronic Noise Sources

If your security boundary contains a PCB

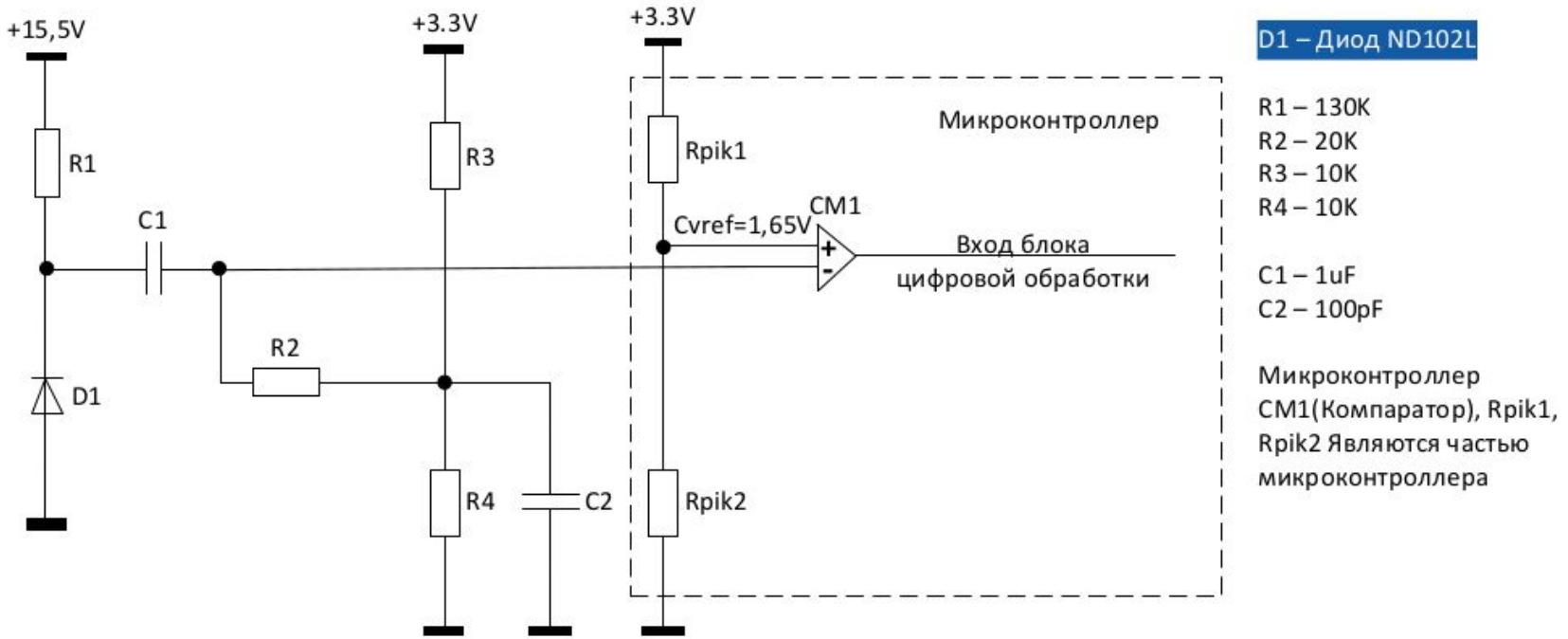


AIS-31 has an example design based on two noisy diodes (avalanche/zener breakdown) and an op amp. A stochastic model is described in [KiSc08].

Other Electronic Noise Sources

A Russian Single (Avalanche) Diode Design

Структурная схема аналогового блока приведена на рисунке 2.



Технические условия на изделие «ГСЧ Гроссмейстер», согласованные с войсковой частью 43753;

Quantum Entropy Sources

A bit more expensive to build (and to physically protect)

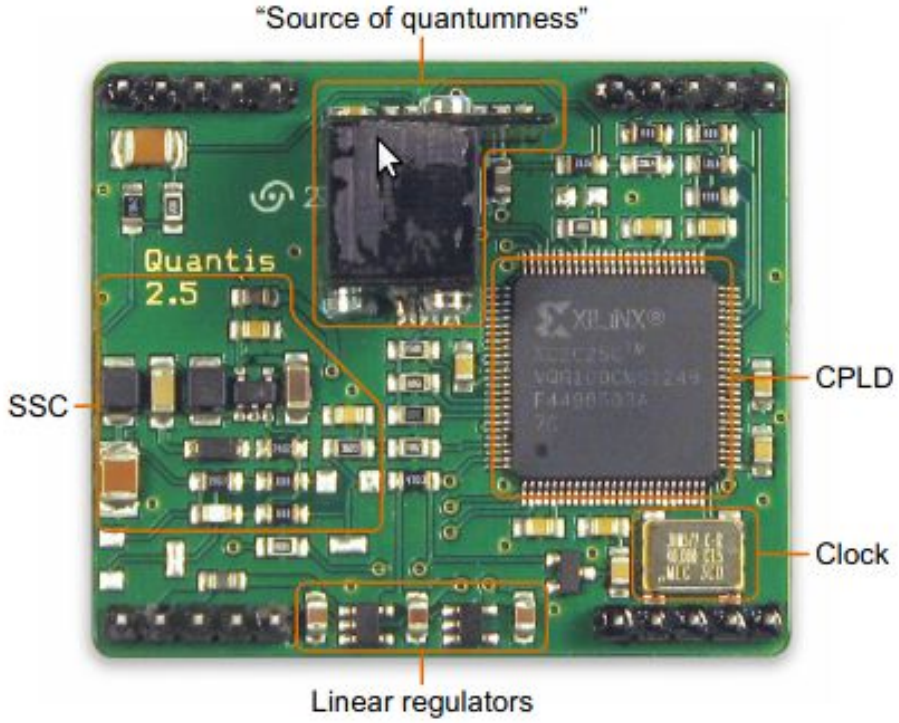


FIG. 1. Main PCB, component side. SSC - step-up switching DC/DC converter, CPLD - complex programmable logic device, Clock - system clock.

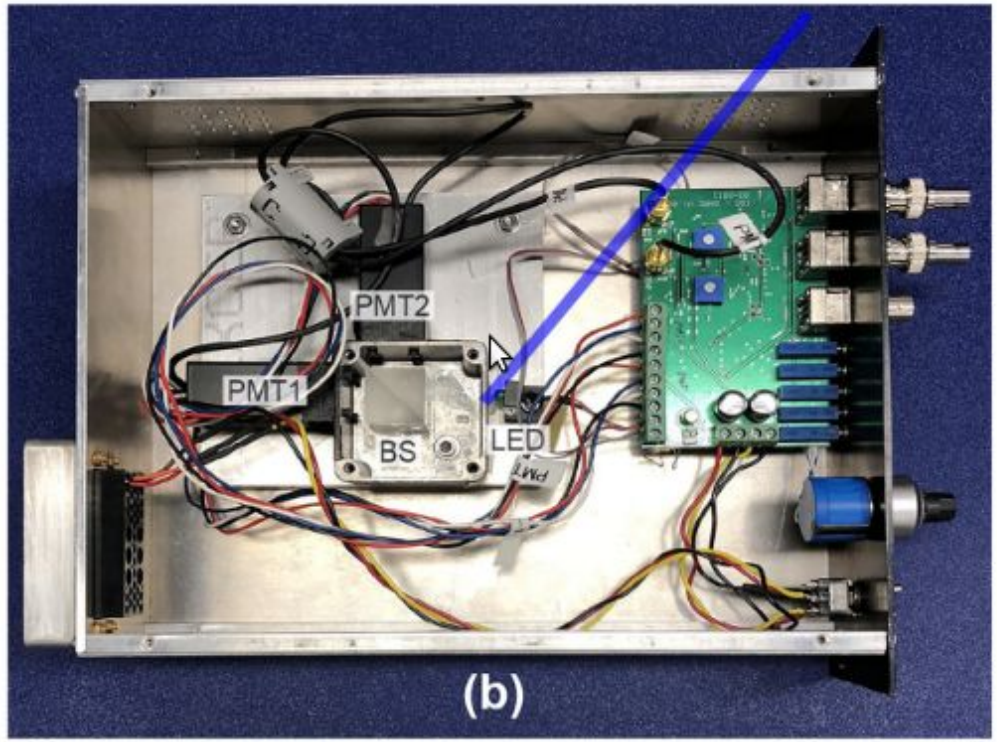


Fig 6. Quantum random number generator under attack. (a) Scheme of the QRNG. Light from a LED (green beam) passes a pinhole and a beam splitter (BS) with two outputs leading to detectors PMT1 and PMT2. For our attack we take advantage of a ventilation hole at the top of the case. With additional light, we can make one detector more likely to click. We show in blue a possible path from the ventilation hole to the pinhole that gives access to the metal box with the BS. (b) Picture of the prototype QRNG [65], with covers removed from the enclosure and beam splitter box. The path to the pinhole has been marked with a blue line.

https://doi.org/10.1371/journal.pone.0236630.g006

Quantum Entropy Sources

Are Photons more “Quantum” than Electrons?

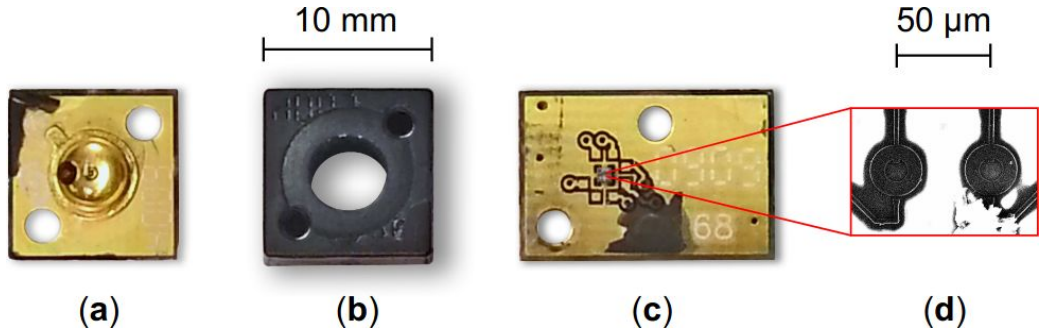


FIG. 2. “Source of quantumness” taken apart. (a) Light emitting diode (LED) light source. (b) Anodized aluminum sleeve. (c) Pair of single-photon detectors. (d) Photosensitive areas of the single-photon detectors (electron-microscope image).

Figures From: M. Petrov, I. Radchenko, D. Steiger, R. Renner, M. Troyer, V. Makarov. *"Independent security analysis of a commercial quantum random number generator"* (2020) https://arxiv.org/abs/2004.04996

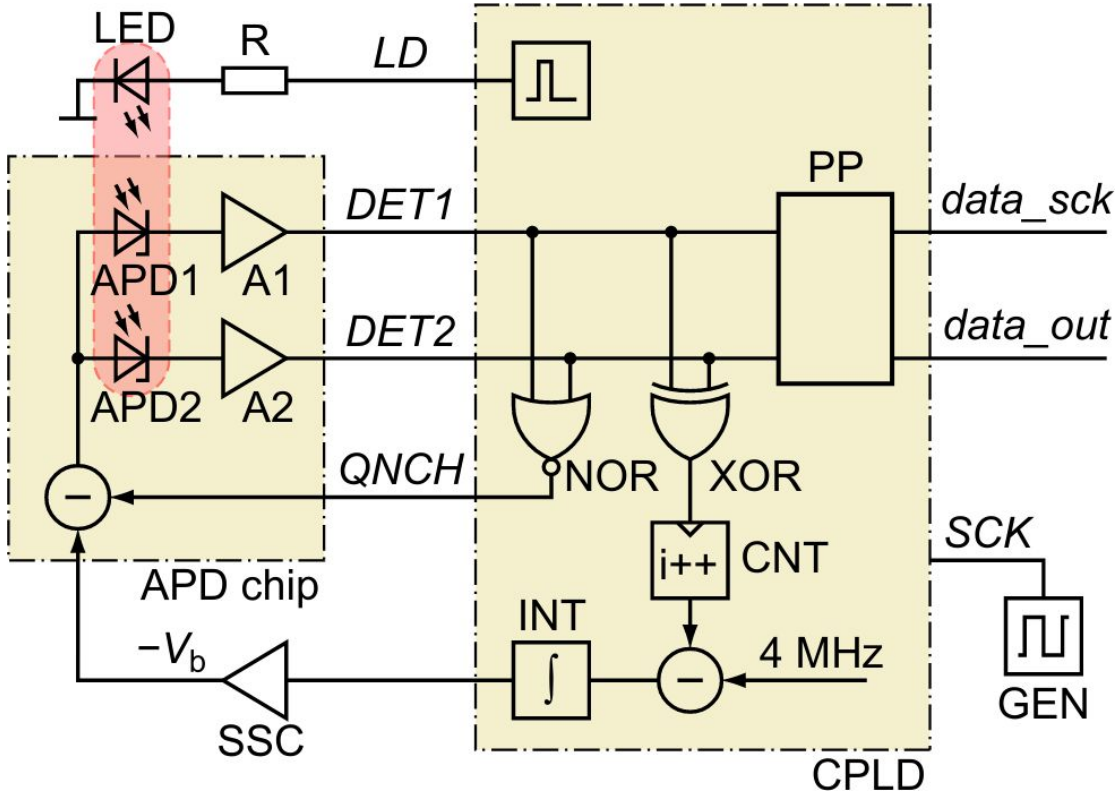
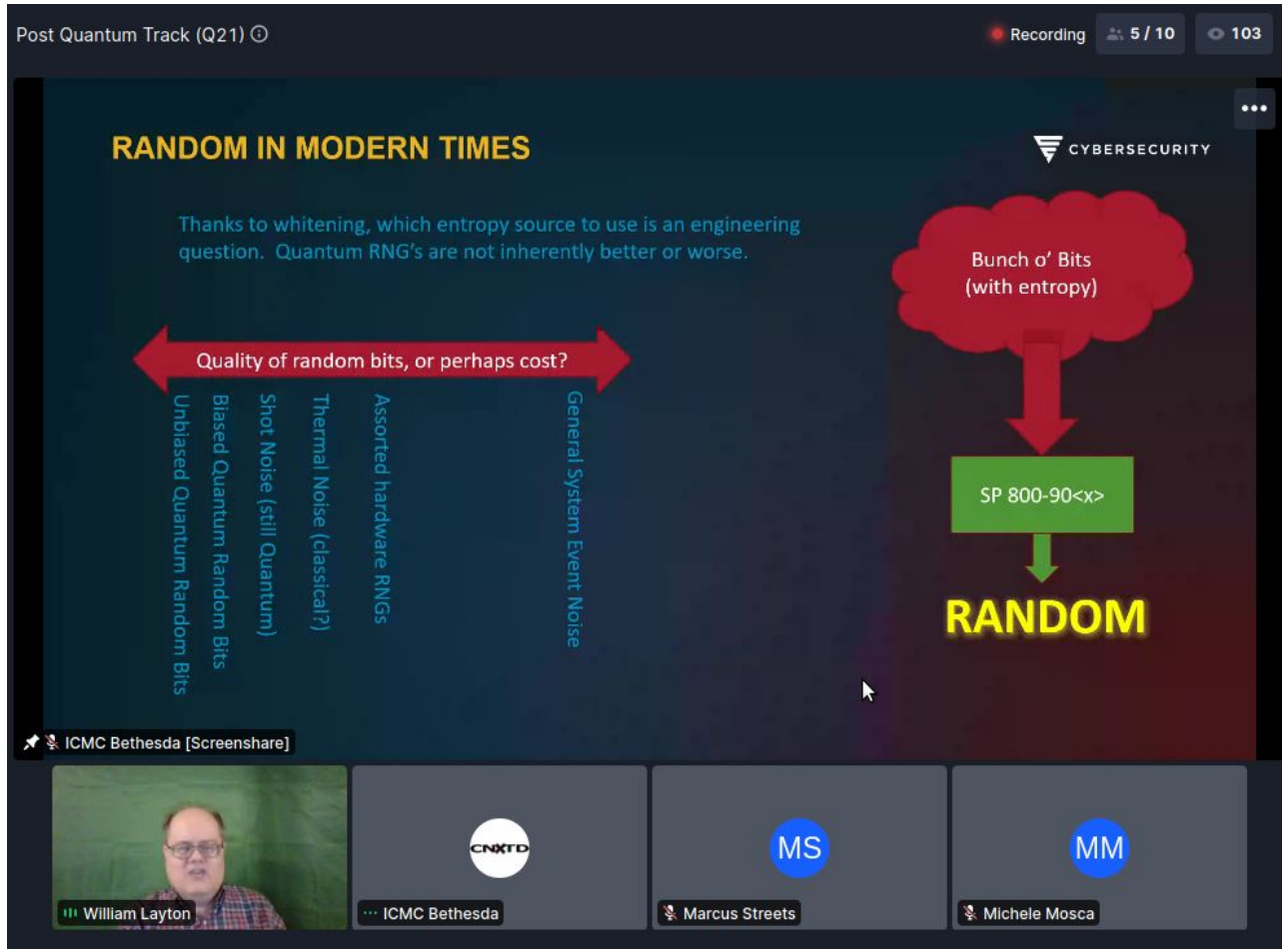


FIG. 3. Simplified electrical scheme of Quantis. A, amplifier; APD, avalanche photodiode; CNT, counter; CPLD, complex programmable logic device; GEN, clock generator; INT, integrator; LED, light emitting diode; NOR, inverted OR gate; PP, post-processing algorithm; R, resistor; SSC, switching power supply; XOR, exclusive OR gate.

Military Requirements

Or “National Security Systems” NSS / CSfC / NIAP



NSS / NIAP testing can use the same SP 800-90x concepts and procedures.

👈 NSA has no significant preference for the type of physical entropy source.

CNSA & Post-Quantum Transition has no impact on random numbers.

Quantum Entropy Sources

NSA and GCHQ emphasize that they don't need QRNGs

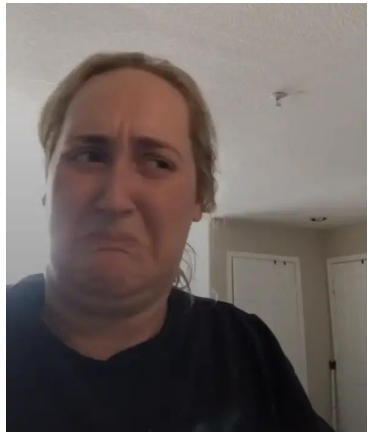
NSA: *"There are a variety of non-quantum RNGs available that have been appropriately validated or certified as acceptable for use in NSS or other government applications. They will remain secure even if a CRQC is built."* (August 2021)
https://www.nsa.gov/Cybersecurity/Post-Quantum-Cybersecurity-Resources/

NCSC/GCHQ: *"The NCSC believes that classical RNGs will continue to meet our needs for government and military applications for the foreseeable future."* (March 2020)
https://www.ncsc.gov.uk/whitepaper/quantum-security-technologies

After the infamous Dual_EC_DRBG generator (that was in SP 800-90A in 2006-2014), it is of course sensible to be suspicious of NSA's random number recommendations. But I haven't heard any good reason to use QRNGs either, but many reasons for not to.

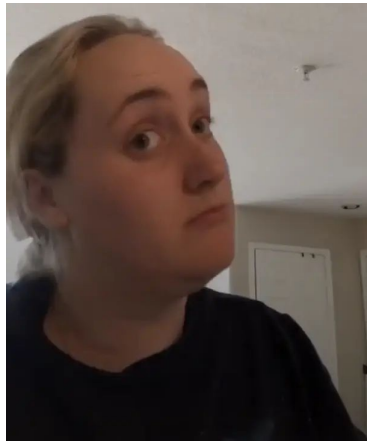
How a Security Engineer Sees It

Module should be Easy to Manufacture, Integrate, and Certify



Bad Signs:

- Nonstandard interfaces, nonstandard algorithms, no theory.
- Obsolete evidence: FIPS 140-2, SP 800-22, or Diehard.
- Generally: Testing of processed output bits with statistical tests.
- Unusual components, expensive manufacturing moves.



Good Signs:

+ Engineered to meet relevant security standard requirements.
+ A stochastic model and a full entropy report available.
+ Has all SP 800-90B required components such as health tests.
+ Ideally: standard cell library RTL source code (if building a SoC).
+ Neat (logical) security boundary and standard interfaces.

RISC-V Entropy ISA

Terminology

- **RISC-V** is an **Instruction Set Architecture** (ISA) in the same sense as ARMv7, ARMv8, i386, or x86 are. Have Compiler & Linux support etc.

- **Open standard**: Many public and commercial designs for RISC-V CPU chips. Many vendors and variants: 32-bit, 64-bit, Vector, FPU, etc.

- Anyone can add *Custom ("X") Extensions*, but for interoperability the **RISC-V International (RVI)** maintains a set of *standard extensions*.

- **Scalar Cryptography & Entropy Source** is a new standard **"K"** extension from the CETG (Cryptography Extensions Task Group).

Get the Spec! It's *free to use.*

September 2021: Frozen

RISC-V Cryptography Extensions Volume I

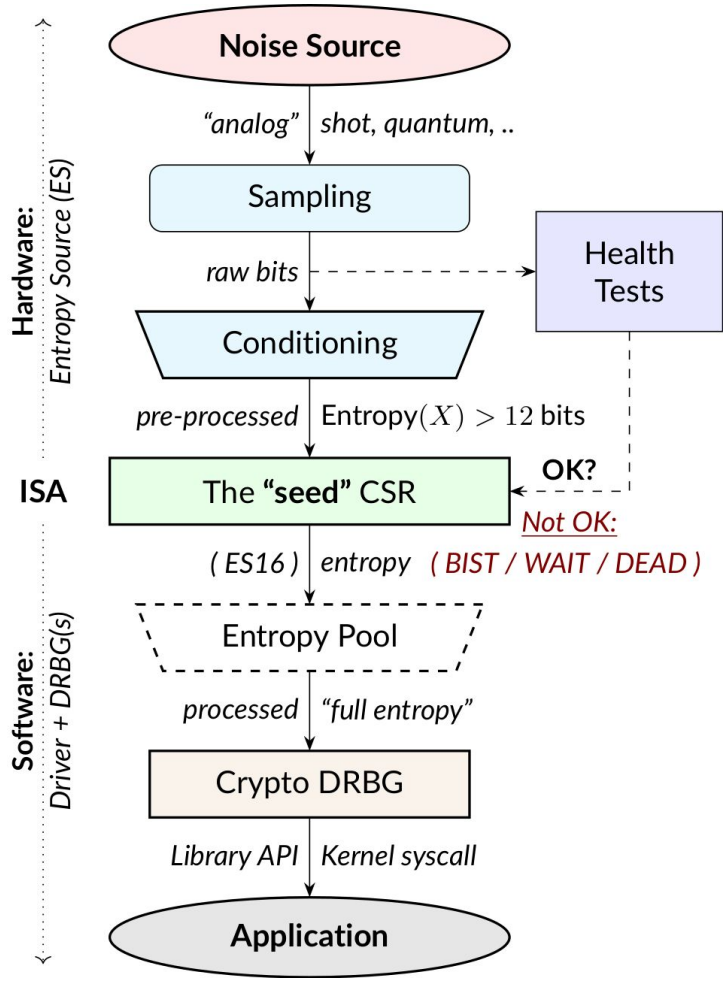
Scalar & Entropy Source Instructions

https://github.com/riscv/riscv-crypto/releases

RISC-V RBG Architecture in Short

The “seed” Control and Status Register (CSR)

- **Not** a PRNG/DRBG like Intel RDRAND!
- An on-chip Physical Entropy Source with >0.75 min-entropy (>0.997 Shannon).
- Requires (2:1) cryptographic conditioning (on driver/sw side) for 90C *“Full Entropy.”*
- Can be used to **seed** a FIPS DRBG / RBG2, RBG3, or by (Linux) Kernel random.
- Both Linux and Microcontroller class.



Accessing the RISC-V Entropy Source

The “seed” Control and Status Register (CSR)

Accessed via control and status register **seed** (CSR 0x015). Assembler:

```
csrrw rd, seed, x0
```

- Instruction returns a word in **rd**, with a 2-bit status + 16 bit entropy.
- Access type must be a *read-write* (for flushing the entropy value).
- CSR is is access controlled with mseccfg.**sseed** and mseccfg.**useed** bits. *M mode may be the only mode in a Microcontroller implementation.*

Polling Randomness from “seed” CSR

High bits **seed[31:30] = OPST** indicate status:

00	**BIST**	Built-In Self-Test.
01	**WAIT**	Sufficient entropy is not yet available.
10	**ES16**	*Success*: Have 16 bits in seed[15:0].
11	**DEAD**	An unrecoverable self-test error.

CSR accesses return immediately, so the interface needs to be repeatedly *polled* (this can be opportunistic -- e.g. in interrupts).

ES16: Return bits always meet specific min-entropy requirements.

Accessing the RISC-V Entropy Source

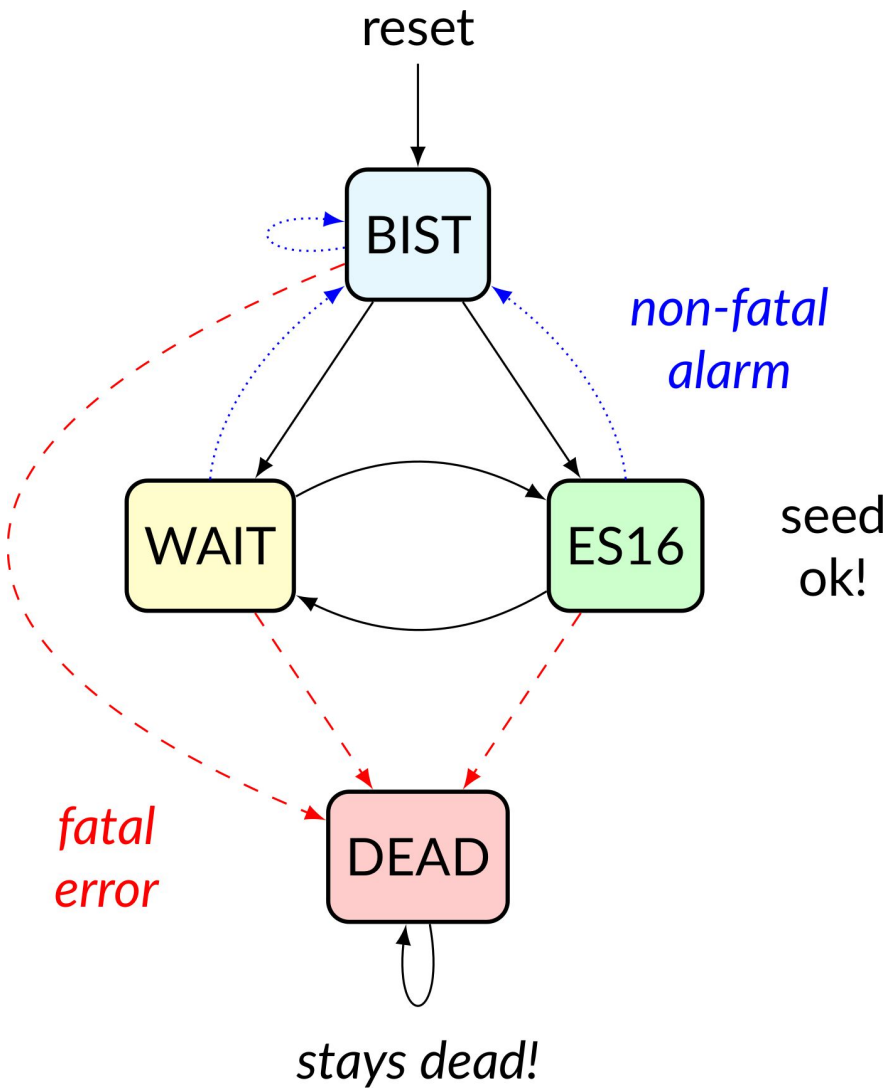
The “seed” Control and Status Register (CSR 0x015)

Bits	Name	Description
31:30	OPST	Status: BIST (00), WAIT (01), ES16 (10), DEAD (11).
29:24	*reserved*	For future use by the RISC-V specification.
23:16	*custom*	Designated for custom and experimental use.
15: 0	entropy	16 bits of randomness, only when OPST=ES16.

Examples:

0x00000000	**BIST** status, no entropy.
0x8000ABCD	**ES16** status, entropy = 0xABCD.
0xFFFFFFFF	**DEAD** status, no entropy.

Polling Randomness from “seed”



- After startup (**BIST**), the state is typically either **WAIT** or **ES16**.
- Ignore **WAIT**, concatenate **ES16**.
- When sufficient amount of entropy is gathered: *Condition (hash) it, and use it to (re)seed a* ***DRBG*** *(SP 800-90A Deterministic Random Bit Generator).*

Implementation Requirements

For successfully polled ES16 “seed” bits

Physical implementations should be of (1) 90B or (2) PTG.2 types.

1. **SP 800-90B IID or non-IID Entropy Source**: 192 bits of assessed min-entropy per 16*16 = 256-bit output block (rate 0.75). With 2:1 driver conditioning this will yield 128 bits of “full entropy”.

2. **BSI AIS-31 PTG.2 Source.** In addition to 90B requirements, also PTG.2 requirements, including 0.997 bits of Shannon entropy.

3. **Virtual machines.** We want emulation to have security too. Host source must have at least PQC Category 5 / 256-bit security level.

Access Control to Physical Sources

Some Security Considerations on Direct Access

Typically this interface is not available in user mode. Supervisor mode Linux kernel may be direct (or trap & emulate if not entirely trusted).

- **Depletion.** Active polling can deny entropy source to all other consumers. Trap & emulate with a DRBG -> harder to deplete.
- **Covert Channels.** Direct physical access may be used to establish communication channels across security boundaries.
- **Hardware Fingerprinting.** Entropy source (and its noise source circuits) may have a uniquely identifiable hardware “signature”.

Implementation Requirements

Health Tests etc

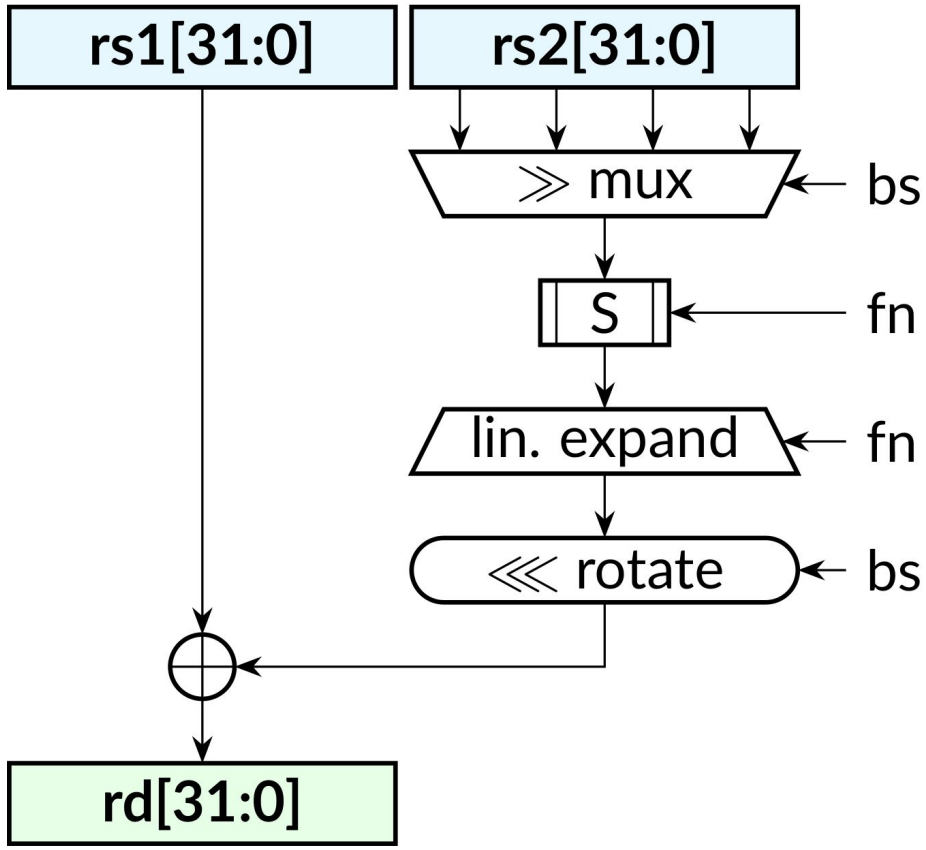
- Like any 800-90B module, the entropy source has mandatory startup and health tests, which may put it in **DEAD** state.
- For AIS.31 some of its startup and online tests may be implemented in software; the driver must be in module certification scope.
- $H \geq n + 64$ with $H=128$, $n=192$ is from Appendix A of draft 800-90C.
- If the ES module can't meet the 0.75 min-entropy rate, either it needs to be reconfigured (e.g. sample rate) or have a better conditioner.

For more rationale, see the spec and https://eprint.iacr.org/2020/866

Implementing the DRBG Drivers

Use the RISC-V Scalar Crypto Extensions!

- The 32-bit and 64-bit AES and SHA instructions in scalar crypto allow *fast, lightweight, constant time AES DRBGs*.
- Reduced overall size and power to having an AES core just for “random”.
- The vector crypto extension will have even higher output speeds for DRBGs.
- Also supports new / non-standard RBGs.



Separate Entropy Validation Scope

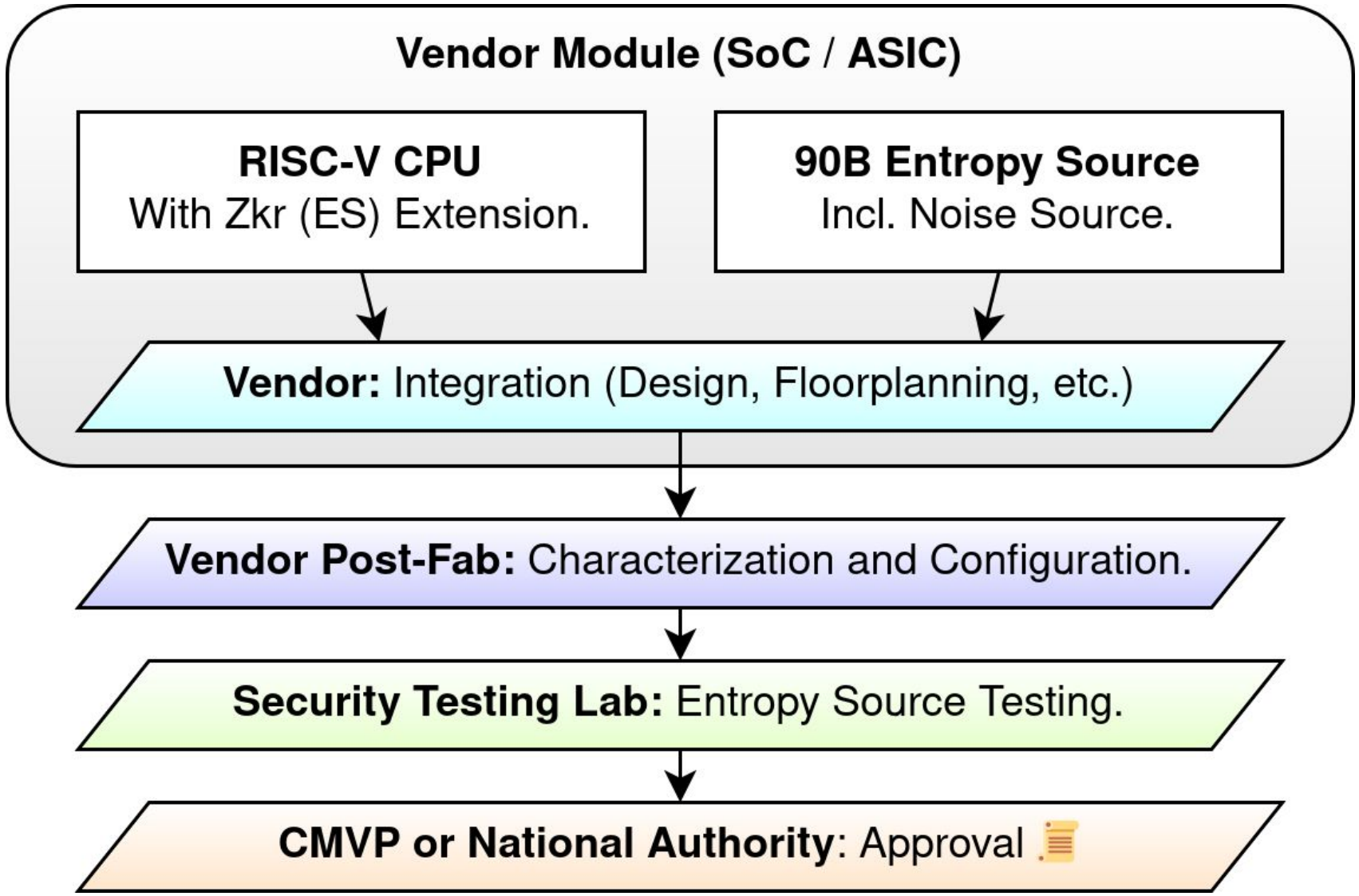
Design reuse little easier for new 90B validations

- (RISC-V) CPUs and Entropy Source modules are licensed by vendors and put into a silicon chip with a bunch of other stuff.

- **Goal**: 90B Entropy Source Module (or AIS-31 RNG IP) can be licensed from a 3rd party and integrated with a RISC-V core.
  -> Thanks to the RISC-V ES Spec, firmware knows what to expect.

- The new *"Entropy Source Validation Scope"* hopefully allows 90B validations to be partially re-used for chips with RISC-V CPUs.

RISC-V + Entropy Source Integration



RISC-V Entropy Source Integration

Checklists 1: Processor Design and Verification

Verify that the RISC-V ISA-defined behaviors are correctly implemented.

Here ES is an abstraction - a *smoke test ES module* may be helpful. Examples:

- ✅ ES16 Entropy words can only be read once (no “peeking” into entropy).
- ✅ Invocation of start-up and online tests on reset or when appropriate.
- ✅ State transitions, especially in case of fatal and non-fatal error signals.
- ✅ Access control in User (U), Supervisor (S), and Machine (M) mode.
- ✅ Custom GetNoise / “mnoise” test interface does not have undue access.

RISC-V Entropy Source Integration

Checklists 2: Noise Source Design and Analysis

The Noise Source Produces the required amount of entropy on the target chip.

- ✅ A stochastic model (or a heuristic argument) that explains why the noise source output is from a random, rather than pseudorandom (deterministic) process.
- ✅ A complete physical model is not necessary, but entropy needs to be justified on the technology node (Semiconductor process? Temperature? Freq? etc.)
- ✅ The noise source is not externally observable (e.g., strongly correlated with the time of day, external power or emissions, or otherwise public or predictable).

Example: For a *ring oscillator* one could show that the source derives an amount of physical entropy from local, spontaneously occurring Johnson-Nyquist thermal noise.

RISC-V Entropy Source Integration

Checklists 3: Entropy Source Module Design

Verify that Entropy Source meets all of the Standard Requirements

A compliant entropy source consists of a noise source *and* additional components whose purpose is to guarantee consistent entropy output.

- ✅ Start-up and health tests. Vendor-defined tests.
- ✅ The conditioner (cryptographic/vetted or non-vetted).
- ✅ A security boundary with clearly defined interfaces. Etc..

One can tabulate all the **SHALL** statements from SP 800-90B, FIPS 140-3 IG (or BSI AIS-31). https://github.com/usnistgov/90B-Shall-Statements

RISC-V Entropy Source Integration

Checklists 4: Characterization and Configuration

Configure source, perform 90B entropy estimation (or AIS-31 statistical tests).

ES modules may have parameters that can be configured at factory (after fabrication).
Example parameters: Ring oscillator sampling Interval, health test α and β bounds.

- ✅ Perform ES Module-defined characterization procedure to derive configuration.
- ✅ Config and parameters must match those submitted to the Security Testing Lab!

Integration testing also replicates the Entropy Source Validation Test System (ESVTS):

- ✅ Use the (IID/Non-IID) estimation algorithms: Record entropy source and raw noise samples using the vendor-defined custom interface. Restart tests, etc.

Suggested GetNoise Test Interface

A Custom CSR location

- Access to raw samples is required for SP 800-90B validation -- the GetNoise conceptual test interface defined in Section 2.3.2 if 90B.

- It is suggested that a custom M-mode **mnoise** CSR = GetNoise().

- *Custom*: Address is vendor-specified. The interface may be permanently disabled in production chips (after factory tests).

- Due to variance in entropy sources, only one CSR bit is defined; NOISE_TEST. However much of the simple testing scripts can be shared between different CPU & ES implementations (open source!)

Suggested GetNoise Test Interface

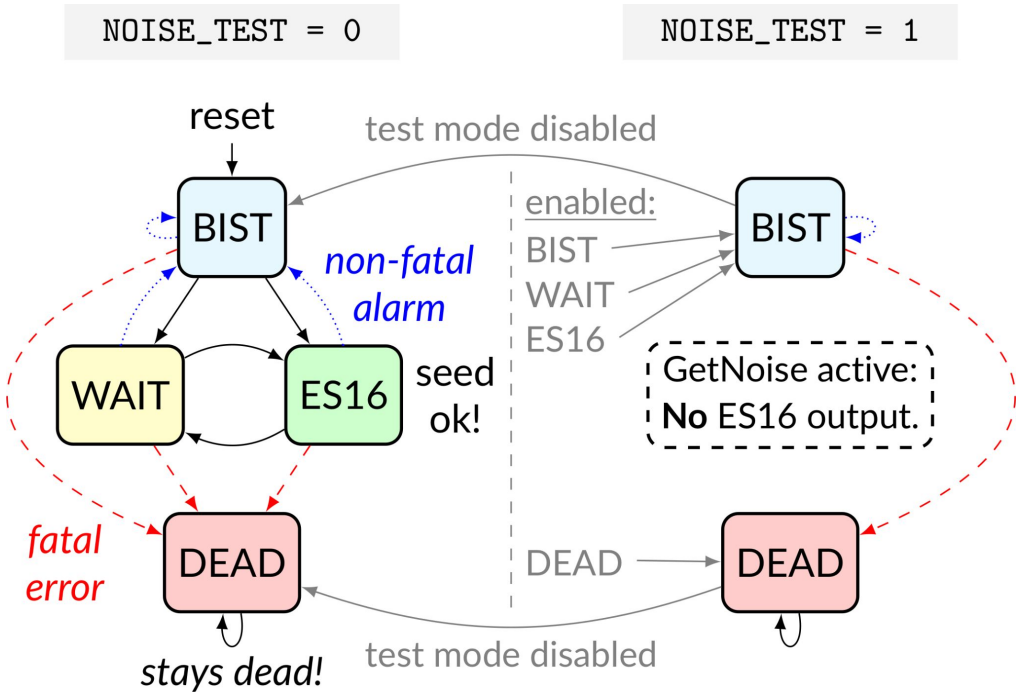
A Custom CSR location

Due to security concerns **seed** CSR entropy is not available while the source is in **mnoise** test mode (*bypassing health checks and conditioning!*)

Testing lab runs scripts that gather datasets (raw, conditioned, startup) and performs estimations.

Note:

The **mnoise** CSR is probably used for the vendor entropy characterization & configuration step too.



Thank You!

Questions..

riscv.org

pqshield.com

Link to specifications:

https://github.com/riscv/riscv-crypto/releases